Fastest Informix DBA Contest II: How Did They Do It?

The winners of the latest contest tuned a process from 40 hours to 1 minute. Here are their techniques.

Lester Knutsen
(lester@advancedatatools.com) is president of Advanced DataTools Corporation, an IBM Informix consulting and training partner specializing in data warehouse development, database design, performance tuning, and Informix training and support. He is president of the Washington, D.C. Area Informix User Group, a founding member of IIUG, an IBM Gold Consultant, and an IBM Data Champion.

Over the summer, we ran another Fastest Informix DBA contest based on our very successful contest at the IIUG Informix Conference in April 2009. We enhanced the rules, made the benchmark process harder, and doubled the size of the database: a customer billing process that had lots of unnecessary SQL, missing indexes, and an ONCONFIG file with some really bad configuration settings. The billing process took 40 hours to complete, and we challenged the participants to make it run faster.

We had more than 70 participants, of whom 8 tuned the process to run in less than 6 minutes. The fastest made it run in a little under 1 minute. And to add to the excitement, only a 5-second difference separated the top three places.

So, congratulations to the new winners! The results were announced at the IBM Information On Demand 2009 Global Conference and updated in a Webcast on November 16. I made a mistake and missed the fastest entry, so we decided to give out two grand prizes. To qualify for the grand prize, the user must be a DBA employed at a company using Informix internally, not a consultant, and not an IBM employee.

- Grand Prize and Fastest Overall Time—Fastest User DBA: Tatiana Saltykova
- Grand Prize—Fastest User DBA: Eric Rowell
- Fastest Consultant: Warren Donovan
- Fastest International DBA: Malte Sukopp, Germany
- Runner-up Consultant: Jeff Filippi
- First Runner-up User DBA: Yunyao (Frank) Qu
- Second Runner-up User DBA: Tammy Frankforter
- Fastest Youngest DBA: Riya Kariath

In this column, I want to highlight what Tatiana, Eric, and Warren did to take a 40-hour process and make it run in 1 minute. They all highlight great examples of what a DBA must do to produce fast code and fine-tuned databases.

Create the right indexes

The database had four tables, each with a primary key, but no other indexes. One of the tricks to database performance is identifying the right number and placement of indexes. Missing indexes will slow down reads, but too many indexes will slow down inserts, updates on indexed fields, and deletes.

The billing process was missing one key index. After the bills were created, the customer table was updated with a new balance, which required the bills table and customer table to be joined (see Figure 1). However, while the customer table had an index on the customer number field because it was the primary key, the bills table did not. Without this index, the only way to find a customer was to do a sequential scan of more than 600,000 bills. Updating 101,000 customers would require 101,000 x 605,280 bills or 61,133,280,000 scans of the bills table. Simply adding this index on the customer number field in the bills table reduced the processing time from 40 hours to about 30 minutes.

```
update customer
    set balance_due = balance_due + ( select sum ( total_bill )
    from bills where bills.customer_number = customer.customer_number )
where customer_number in ( select customer_number from bills );
```

Figure 1: *SQL to update a customer balance—requires an index on customer_number*

```
CREATE TRIGGER ins_bills insert on bills
   REFERENCING NEW AS n
   FOR EACH ROW
   (UPDATE customer
     SET balance_due = balance_due + n.total_bill
    WHERE customer_number = n.customer_number);

BEGIN WORK;
LOCK TABLE bills IN EXCLUSIVE MODE;

--------------------------------------------------------------------
Create bills
--------------------------------------------------------------------
insert into bills
      (
      customer_number,
      last_name,
      …
      total_bill
      )
select
      customer.customer_number,
      customer.last_name,
      …
      product.product_price,
      CASE
            WHEN customer.start_date <= "01/01/2009"
                  AND customer.balance_due > 50000
            THEN 10
            ELSE 0
      END,
      state.sales_tax,
      CASE
            WHEN customer.start_date <= "01/01/2009"
                  AND customer.balance_due > 50000
            THEN ((product_price - 10 ) * (1 + state.sales_tax))
            ELSE (product_price * (1 + state.sales_tax))
      END
from customer, state, product
where customer.state = state.state
and customer.product_code[1] = product.product_code[1]
and product.product_number in (1, 2, 4, 7, 9, 10);
```

Figure 2: *Eric Rowell's Fastest Informix DBA SQL. This SQL takes three inserts and two update statements and optimizes them into one SQL statement and one trigger, increasing performance.*

It's also important to add only necessary indexes. Several participants created additional indexes that did not help the Informix SQL optimizer and actually slowed down the process.

Optimize the SQL statements

The benchmark process had five SQL statements: three inserts into the bills table and two update statements—one to calculate the total bill discount, and one to calculate the new balance. Eric optimized the SQL to a single statement by using a trigger that updated the customer table only when a bill was inserted into the bills table (see Figure 2). This is very efficient code and one of the reasons he performed the task so fast. Eric eliminated the bill discount UPDATE by adding two SQL CASE statements to the INSERT statement. Meanwhile, the trigger on the bills table made the new balance UPDATE statement unnecessary because the customer balance was updated anytime a bill was inserted.

Eric's trigger was an especially brilliant approach because it also eliminated the need for the index on the customer number field in the bills table previously discussed. He eliminated the time to create and maintain this index because he did not need it, and he got the job done faster.

The key to optimizing your SQL is to reduce the number of statements that read through tables. The baseline SQL in this contest read the customer table three times, one for each of the INSERT statements. And, it read the customer table and the bills table two more times for each of the updates. By reducing the SQL to one read and a trigger that immediately updated the customer table, Eric cut the disk I/O and database reads and writes to one-fifth of the baseline.

Reduce the disk I/O

Another technique that both Warren and Eric used was to reduce the number of disk reads. Both used a new feature of Informix: creating dbspaces with a 16 KB page size instead of the default 2 KB page size. This meant that the database engine could gather eight times more data in one read, which benefits most indexes by putting more index data on a single page. The benefit lies in reducing the number of time-consuming disk reads to get all the data into memory. Warren and Eric both set up most of their buffer pools for the 16 KB pages, so this optimized the amount of work that could be done in memory.

More to come at the 2010 IIUG Informix Conference

The contest was a lot of fun to run and monitor, and it is exciting to see the ingenuity and creativity that all the Informix DBAs put into it. Congratulations to all the winners, who are listed on our Web site: www.advancedatatools.com/Informix/index.html.

We will sponsor the next version of this contest, the Fastest Informix DBA Contest III, at the IIUG Informix Conference in Overland Park, Kansas, April 25–28. Visit the Advanced DataTools Web site above for more details or the International Informix User Group Web site at www.iiug.org/conf. Hope to see you there.